*J. Symbolic Computation* (1992) **14**, 31-50

# Computation of Hilbert Functions

DAVE BAYER
*Department of Mathematics*
*Barnard College*
*New York, NY 10027 USA*

MIKE STILLMAN
*Department of Mathematics*
*Cornell University*
*Ithaca, NY 14853 USA*

*(Received 20 December 1990)*

**We present an algorithm along with implementation details and timing data for computing the Hilbert function of a monomial ideal. Our algorithm is often substantially faster in practice than existing algorithms, and executes in linear time when applied to an initial monomial ideal in generic coordinates. The algorithm generalizes to compute multi-graded Hilbert functions.**

## 1. Introduction

Let $k$ be a field, and let $S = k[x_0, \ldots, x_n]$ be a graded polynomial ring, where each $x_i$ is homogeneous of degree one. Let $I$ be a homogeneous ideal in $S$. We consider the problem of computing the Hilbert function of $S/I$, in the case where $I$ is generated by monomials.

Given the data of an arbitrary finitely generated $S$-module $M$, the construction of a Gröbner basis for its presentation matrix is the single most fundamental operation which can be carried out by a computer algebra system (Buchberger, 1965, 1985; Bayer, 1982). From the leading terms of such a basis, one obtains an $S$-module of the form $S/I_1 \oplus \ldots \oplus S/I_r$, where each $I_i$ is a monomial ideal. This $S$-module has the same Hilbert function as $M$, a relationship which was first observed by Macaulay (1927). Thus via Gröbner bases, one reduces the computation of Hilbert functions for $S$-modules $M$ to the case considered here.

In this paper, we present an algorithm for computing the Hilbert function of a monomial ideal, and give implementation details and timing data for this algorithm.

Several algorithms for computing Hilbert functions in this case are presented in Mora and Möller (1983). Our algorithm, which can be significantly faster in practice, has been influenced by their ideas, and by computational experience with the computer algebra system *Macaulay* (Bayer & Stillman 1982-1992), which we have been developing over most of the last decade. The implementation described here will migrate soon to *Macaulay*, and represents an advance on the algorithm employed by the version of *Macaulay* available as of this writing.

0747-7171/92/070031+20 $03.00/0 © 1992 Academic Press Limited

Hilbert functions have become a bottleneck only in the largest Gröbner basis computations now possible. As the capabilities of machines and computer algebra systems continue to advance, it will become increasingly imperative to adopt improved algorithms for this subproblem. This paper is likely to be of particular interest to designers of computer algebra systems.

## 2. Hilbert Functions

Recall that the Hilbert function of the finitely generated $S$-module $M$ is defined to be the function

$$p(d) = \dim M_d,$$

where $M_d$ is the degree $d$ part of $M$ (See Atiyah & Macdonald (1969, Chapter 11) or Stanley (1978)).

The Hilbert polynomial $q(d)$ is the unique polynomial in $d$ such that $q(d) = p(d)$ for all $d \gg 0$. Let $X \subset \mathbf{P}^n$ be the support of $M$; $q(d)$ has the form

$$q(d) = \frac{e}{r!}d^r + \cdots$$

where $e$ is the degree of $X$, and $r$ is its dimension.

Recall also that the Hilbert series, or Poincaré series, of $M$ is defined to be the generating function

$$H_M(t) := \sum_{d=-\infty}^{\infty} p(d)t^d = \frac{g(t)}{(1-t)^{n+1}},$$

where $g(t)$ is a Laurent polynomial in $t$. This representation of the Hilbert function is suitable for manipulating by computer. Throughout this paper, we use "computing the Hilbert function" and "computing the Hilbert series" interchangeably. It is easy to compute the Hilbert polynomial from the Hilbert series, by expanding it as a Laurent series in $(1-t)$:

$$\frac{g(t)}{(1-t)^{n+1}} = \frac{a_{-r-1}}{(1-t)^{r+1}} + \cdots + \frac{a_{-1}}{(1-t)} + e(t),$$

where $e(t)$ is a Laurent polynomial in $t$. We can now read off the Hilbert polynomial $q(d)$ from the polar part of this expansion, since

$$\sum_{d=0}^{\infty} \binom{j+d}{j} t^d = \frac{1}{(1-t)^{j+1}}$$

is the Hilbert series of $\mathbf{P}^j$, for each $j$. We have

$$q(d) = a_{-r-1}\binom{r+d}{r} + \cdots + a_{-2}\binom{1+d}{1} + a_{-1}.$$

Thus, the polar part of the generating function for $p(d)$ carries the information which persists in all degrees. The polynomial part gives the error between $p(d)$ and $q(d)$ in a

finite number of degrees:

$$e(t) = \sum_{d=-\infty}^{\infty} (p(d) - q(d))t^d.$$

Finding the Hilbert series of arbitrary graded $S$-modules can be reduced via Gröbner bases to the computation of Hilbert series of monomial ideals. Recall that $S(-d)$ is the $S$-module $S$ with the shifted grading $S(-d)_e = S_{-d+e}$. Write $M = F/L$, where $F = \bigoplus_{j=1}^m Se_j \cong \bigoplus_{j=1}^m S(-d_j)$ is a graded free $S$-module, $\deg e_j = d_j$, and $L \subset F$ is a graded submodule.

Let $>$ be a total order on the monomials $\{\mathbf{x}^A e_i\}$ of $F$, satisfying $\mathbf{x}^A e_i > \mathbf{x}^B e_j \Rightarrow \mathbf{x}^C \mathbf{x}^A e_i > \mathbf{x}^C \mathbf{x}^B e_j$, for all monomials $x^C$ of $S$, and satisfying $x_i e_j > e_j$, for each $j$. Such an order is called a multiplicative order on $F$. For $f \in F$, define $\text{in}(f)$ to be the initial (greatest) term of $f$. For $L \subset F$, define the initial submodule $\text{in}(L) := (\text{in}(f) \mid f \in L)$. $\text{in}(L)$ is most easily computed using Gröbner bases (Buchberger, 1965, 1985; Bayer, 1982).

$L$ and $\text{in}(L)$ have the same Hilbert series by the following immediate generalization of a theorem of Macaulay (1927).

**Proposition 2.1** *If $L \subset F = \bigoplus_{j=1}^m Se_j$ is a graded submodule, then for each $d \in \mathbf{Z}$,*

$$\dim L_d = \dim \text{in}(L)_d. \quad \square$$

If we define $I_j e_j := \text{in}(L) \cap Se_j$, for $j = 1, \ldots, m$, then each $I_j \subset S$ is a monomial ideal, and $F/\text{in}(L) \cong S/I_1 \oplus \ldots \oplus S/I_m$. Therefore

$$\langle L \rangle = \sum_{j=1}^m t^{d_j} \langle I_j \rangle,$$

where $\langle L \rangle$ denotes the numerator of the Hilbert series of $F/L$ and $\langle I_j \rangle$ denotes the numerator of the Hilbert series of $S/I_j$. In either the ideal or submodule case, the problem of computing the Hilbert series reduces to the computation of the Hilbert series of a monomial ideal. Below, we give an algorithm for computing the Hilbert series of a monomial ideal which we believe is the fastest to date.

See Mora & Möller (1983) for earlier work on computing Hilbert polynomials.

We now develop a number of rules, to be used in recursively computing Hilbert series.

**Proposition 2.2** *Let $I$ be a monomial ideal, and write $I = (J, x^A)$ for a monomial ideal $J$ and a monomial $x^A$. Let $\langle I \rangle$ denote the numerator $g(t)$ of the Hilbert series for $S/I$, and let $|A|$ denote the total degree of the monomial $x^A$. Then*

*(a)* $\langle x^A \rangle = 1 - t^{|A|}$;

*(b)* $\langle J \cap (x^A) \rangle = 1 - t^{|A|} + t^{|A|} \langle J : x^A \rangle$;

*(c)* $\langle I \rangle = \langle J \rangle - t^{|A|} \langle J : x^A \rangle$.

**Proof.** (a) We want to count the monomials of $S/(x^A)$, i.e. the monomials of $S$ which don't belong to $(x^A)$. $1/(1-t)^{n+1}$ counts the monomials of $S$. $\dim(x^A)_d = \dim S_{d-|A|}$

for each $d$, because $(x^A) \cong S(-|A|)$, so $t^{|A|}/(1-t)^{n+1}$ counts the monomials of $(x^A)$. Thus $(1-t^{|A|})/(1-t)^{n+1}$ counts the monomials of $S/(x^A)$.

(b) The monomials of $S/(J \cap (x^A))$ fall into two groups: those that lie outside of $(x^A)$, and those that belong to $(x^A)$ but lie outside of $J \cap (x^A)$. The first group of monomials is counted by $\langle x^A \rangle$. $(J \cap (x^A)) = x^A(J : x^A)$, so $(x^A)/(J \cap (x^A)) \cong (S/(J : x^A))(-|A|)$, and this second group of monomials is counted by $t^{|A|}\langle J : x^A \rangle$.

(c) We have

$$\langle I \rangle = \langle J, x^A \rangle = \langle J \rangle + \langle x^A \rangle - \langle J \cap (x^A) \rangle,$$

since the monomials outside of $(J, x^A)$ can be counted by adding the monomials outside of $J$, the monomials outside of $(x^A)$, and subtracting the monomials outside of $J \cap (x^A)$. Now substitute (a) and (b). □

**Corollary 2.3** *If $I = (x^{A_1}, \ldots, x^{A_r}) \subset S$ is a monomial ideal, then*

$$\langle I \rangle = \langle x^{A_1} \rangle - \sum_{i=2}^{r} t^{|A_i|} \langle x^{A_1}, \ldots, x^{A_{i-1}} : x^{A_i} \rangle.$$

**Proof.** Apply Proposition 2.2(a,c), and induction on $r$. □

**Proposition 2.4** *Let $I$ be a monomial ideal. Suppose that the variables $x_0, \ldots, x_n$ of $S$ can be partitioned into disjoint sets $V_1 \cup \cdots \cup V_j$, such that each generator of $I$ belongs to the subring $k[V_i]$ for some $i$. Define $I_i = I \cap k[V_i]$. Then*

$$\langle I \rangle = \prod_{i=1}^{j} \langle I_i \rangle.$$

**Proof.** We have the tensor product decomposition

$$S/I = k[V_1]/I_1 \otimes_k \cdots \otimes_k k[V_j]/I_j,$$

and Hilbert series multiply with respect to tensor products. □

We may compute $\langle I_i \rangle$ using $I_i S \subset S$ for the same reason: $S/I_i S$ is the tensor product of $k[V_i]/I_i$ with the remaining variables.

**Corollary 2.5** *Let $I$ be a prime monomial ideal, of codimension $r$. Then*

$$\langle I \rangle = (1-t)^r.$$

**Proof.** $I$ is generated by $r$ degree one monomials, so $\langle I \rangle = \langle x_i \rangle^r$. Now apply Proposition 2.2(a). □

The following algorithm for computing Hilbert series works by recursively applying Proposition 2.2(a),(c):

**Algorithm 2.6 (Hilbert series)**

input:
A monomial ideal $I$ minimally generated by $(x^{A_1}, \ldots, x^{A_r})$.
output:
The numerator $\langle I \rangle$ of the Hilbert series for $S/I$
begin
(*a) rearrange $(x^{A_1}, \ldots, x^{A_r})$ so that they are in ascending
lexicographic order on the *reversed* set of variables $x_n, \ldots, x_0$.
(b) if $r = 0$
then set $h(t) := 1$
else set $h(t) := 1 - t^{|A_1|}$.
(c) for $i := 2$ to $r$ do
define $x^{B_j}$ to be the least monomial such that $x^{A_i}x^{B_j}$
is a multiple of $x^{A_j}$, for $j = 1, \ldots, i-1$.
set $J = (x^{A_1}, \ldots, x^{A_{i-1}} : x^{A_i}) := (x^{B_1}, \ldots, x^{B_{i-1}})$,
and reduce to a minimal set of generators.
Compute $\langle J \rangle$ by one of the following three variants.
*variant A:*
set $(J_1, x_{j_1}, \ldots, x_{j_k}) := J$, where
$J_1$ contains no linear monomials .
compute $\langle J_1 \rangle$ by a recursive call to this procedure.
set $\langle J \rangle := (1-t)^k \langle J_1 \rangle$.
*variant B:*
partition the variables $x_0, \ldots, x_n$ into disjoint sets
$V_1 \cup \ldots \cup V_p$ such that each generator of $J$
belongs to the subring $k[V_j]$, for some $j$.
for each $j = 1, \ldots, p$ do
set $J_j := J \cap k[V_j]$
compute $\langle J_j \rangle$ by a recursive call to this procedure.
end for.
set $\langle J \rangle := \langle J_1 \rangle \langle J_2 \rangle \ldots \langle J_p \rangle$.
*variant C:*
compute $\langle J \rangle$ by a recursive call to this procedure.
set $h(t) := h(t) - t^{|A_i|} \langle J \rangle$.
end for.
return $\langle I \rangle := h(t)$.
end.

Step (a) is optional, and one of the variants (A), (B), or (C) must be chosen. Step (a) and variant (A) are very easy to implement and have a pivotal effect on the running time. One would expect variant (B) to give a performance increase, but this doesn't appear for small to moderate size problems, because of the overhead of partitioning. Variant (C) is included as a base algorithm; it is significantly slower than the other variants in the case when the number of variables is large.

**Proposition 2.7** *Algorithm 2.6 correctly computes $\langle I \rangle$, whichever variant (A), (B) or (C) is chosen, and whether or not the optional step (a) is omitted.*

**Proof.** The algorithm terminates, since each recursive procedure call has as its argument an ideal with fewer generators than $I$, and the case of zero or one generator is handled separately in step (b). No subsequent step depends on step (a) for correctness. Assuming that each variant (A), (B), and (C) correctly computes $\langle J \rangle$, the validity of the for loop (c) follows from Corollary 2.3.

Variant (A) correctly computes $\langle J \rangle$ by Proposition 2.4 and Corollary 2.5. The validity of variant (B) follows from Proposition 2.4. If variant (C) is chosen, $\langle J \rangle$ is set correctly by a recursive procedure call. □

**Proposition 2.8** *Let $T(m,n)$ denote an upper bound for the number of recursive procedure calls made by Algorithm 2.6, to compute the Hilbert series of a monomial ideal $I$ with $m$ generators in the $n+1$ variables of $\mathbf{P}^n$. Then*

*(i) if step (a) is omitted, we can take*

$$T(m,n) = 2^{m-1};$$

*(ii) if step (a) is not omitted, we can take*

$$T(m,n) = \begin{cases} \binom{m}{n} + \binom{m}{n-2} + \ldots + \binom{m}{0} & \text{if } n \geq 0 \text{ is even} \\ \binom{m}{n} + \binom{m}{n-2} + \ldots + \binom{m}{1} & \text{if } n \geq 1 \text{ is odd,} \end{cases}$$

*for any $m \geq 1$, so for $m \gg n$,*

$$T(m,n) \approx \frac{m^n}{n!}.$$

**Proof.** (i) In this case, $T(m,n)$ satisfies the initial condition $T(1,n) = 1$ (we are counting the original procedure call), and the recurrence $T(m,n) = T(m-1,n) + \ldots T(1,n) + 1$. Therefore one can take $T(m,n) = 2^{m-1}$.

(ii) If $x^{A_i}$ is greater in the given lexicographic order than any of the monomials $x^{A_1}, \ldots, x^{A_{i-1}}$, then none of the generators $x^{C_1}, \ldots, x^{C_m}$ of $(J : x^A)$ involve the last variable $x_n$. Thus each recursion via variant (C) drops a variable. Since every ideal in $k[x_0]$ is principal, $\langle I \rangle$ will be computed without recursion by step (b) whenever the argument $I$ involves only the variable $x_0$. Thus $T(m,n)$ satisfies the initial conditions $T(1,n) = T(m,0) = 1$, and the recurrence $T(m,n) = T(m-1,n-1) + T(m-2,n-1) + \ldots + T(1,n-1) + 1$. Using the fomula

$$\binom{1}{d} + \ldots + \binom{m-1}{d} = \begin{cases} \binom{m}{1} - 1 & \text{if } d = 0 \\ \binom{m}{d+1} & \text{if } d \neq 0 \end{cases},$$

one checks that the given expression for $T(m,n)$ satisfies these rules. □

Clearly, (ii) holds no matter what order we choose for the variables $x_0, \ldots, x_n$ in step (a). Our purpose in reversing the variables will become clear after Proposition 2.10.

In practice, this algorithm will run much faster than these bounds indicate, since the ideal quotient $(x^{A_1}, \ldots, x^{A_{i-1}} : x^{A_i})$ will usually have fewer than $i-1$ generators.

One might wonder whether there is an algorithm for Hilbert functions which executes in polynomial time in the size of the input. The following proposition shows that such an algorithm cannot exist (unless $P = NP$). The proof will be readily apparent to readers familiar with NP arguments; we include it for completeness.

**Proposition 2.9** *The following problem is NP-complete: Given a monomial ideal* $J \subset k[x_0, \ldots, x_n]$, *and an integer* $K$, *is the codimension of* $J \leq K$?

**Proof.** Recall the NP-complete problem VERTEX COVER (Karp (1972), Garey & Johnson (1979)): Given a graph $G = (V, E)$ and a positive integer $K \leq |V|$, is there a subset $V' \subset V$ such that $|V'| \leq K$, and for each edge $\{u, v\} \in E$, at least one of $u$ and $v$ belongs to $V'$? Such subsets are called *vertex covers.*

Associate the polynomial ring $k[x_0, \ldots, x_n]$ with the vertex set $V = \{0, \ldots, n\}$. For each subset $V' \subset V$, define $P(V')$ to be the prime ideal $(x_i \mid i \in V')$. $P(V')$ defines a linear space of codimension $|V'|$. A monomial ideal $J$ has codimension $\leq K$ if and only if $J \subset P(V')$ for some $V'$ with $|V'| \leq K$.

If $J$ has codimension $\leq K$, then given an appropriate choice of $V'$, one can quickly confirm that $J \subset P(V')$ and $|V'| \leq K$. Thus, our problem belongs to the class NP.

Given a graph $G = (V, E)$, define $J$ to be the monomial ideal $(x_i x_j \mid \{i, j\} \in E)$. $V'$ is a vertex cover of size $\leq K$ if and only if $J$ has codimension $\leq K$. Thus VERTEX COVER is reducible to our problem, establishing NP-completeness. □

There is an important class of monomial ideals, the *Borel* ideals, for which the above algorithm completes after linearly many steps. In fact, a closed form solution can be determined. A monomial ideal $I$ is defined to be *Borel* if $x_j x^B \in I$ implies that $x_i x^B \in I$ for all $i < j$.

**Proposition 2.10** *Let* $I = (x^{A_1}, \ldots, x^{A_r}) \subset S = k[x_0, \ldots, x_n]$ *be a Borel monomial ideal. Let* $d_i = \deg x^{A_i}$, *and let* $c_i$ *be the largest index* $j$ *such that* $x_j$ *divides* $x^{A_i}$. *If* $\{x^{A_1}, \ldots, x^{A_r}\}$ *minimally generates* $I$, *then the numerator* $\langle I \rangle$ *of the Hilbert series for* $S/I$ *is*

$$\langle I \rangle = 1 - \sum_{i=1}^{r} t^{d_i} (1-t)^{c_i}.$$

**Proof.** Rearrange the $x^{A_i}$ as in the algorithm so that $x^{A_{i+1}}$ is greater than $x^{A_i}$, for all $i$, using the lexicographic order $x_n > x_{n-1} > \ldots > x_0$. This ensures that any initial subsequence $\{x^{A_1}, \ldots, x^{A_q}\}$ also minimally generates a Borel ideal. Notice also that

$$(x^{A_1}, \ldots, x^{A_{r-1}} : x^{A_r}) = (x_0, \ldots, x_{c_i - 1}),$$

by the Borel property. Therefore, by Corollary 2.3,

$$\langle I \rangle = \langle x^{A_1} \rangle - \sum_{i=2}^{r} t^{d_i} \langle x_0, \ldots, x_{c_i - 1} \rangle.$$

The desired formula now follows from Corollary 2.5. □

For an alternate approach to this problem, see Eliahou & Kervaire (1990). They compute explicit finite free resolutions for these ideals. For a somewhat simpler, Gröbner basis approach to determine finite free resolutions of Borel ideals, see Bayer & Stillman (1992).

In characteristic zero, and in characteristic greater than the degrees of generators of $I$, $I$ is Borel if and only if it is fixed by the Borel subgroup of $GL(n+1)$. This will be the case for initial ideals in generic coordinates; see Galligo (1974), Bayer & Stillman (1987a, 1987b). Moreover, initial ideals in nongeneric coordinates often *nearly* satisfy this condition, so if the monomials are sorted in ascending lexicographic order on the reversed variables $x_n > x_{n-1} > \ldots > x_0$, then the ideal quotients of Algorithm 2.6 will partition in many of the recursive steps of variants (A) or (B). Thus, one can expect these algorithms to be fast in practice, even on very large examples.

## An Example: The Rational Quartic Curve in $\mathbf{P}^3$

Let $S = k[a,b,c,d]$, and let $I = (ac-bd, ab^2-c^3, a^2b-c^2d, a^3-cd^2)$. This ideal defines a rational quartic curve in $\mathbf{P}^3$. With respect to the lexicographic order (on $a,b,c,d$), a Gröbner basis for $I$ is given by $(ac-bd, ab^2-c^3, a^2b-c^2d, a^3-cd^2, b^3d-c^4)$. Thus $\mathrm{in}(I) = (ac, ab^2, a^2b, a^3, b^3d)$. Arranging the generators of $\mathrm{in}(I)$ in increasing lexicographic order on the reversed variables $d,c,b,a$, we have $\mathrm{in}(I) = (a^3, a^2b, ab^2, ac, b^3d)$. We follow variant (A) to compute $\langle \mathrm{in}(I) \rangle$:

$$\begin{aligned}
&\langle a^3, a^2b, ab^2, ac, b^3d \rangle \\
&\quad = \langle a^3 \rangle - t^3\langle a^3 : a^2b \rangle - t^3\langle a^3, a^2b : ab^2 \rangle - t^2\langle a^3, a^2b, ab^2 : ac \rangle \\
&\qquad - t^4\langle a^3, a^2b, ab^2, ac : b^3d \rangle \\
&\quad = 1 - t^3 - t^3\langle a \rangle - t^3\langle a \rangle - t^2\langle a^2, ab, b^2 \rangle - t^4\langle a \rangle \\
&\quad = 1 - t^3 - t^3(1-t) - t^3(1-t) - t^2\langle a^2, ab, b^2 \rangle - t^4(1-t) \\
&\quad = 1 - 3t^3 + t^4 + t^5 - t^2\left(\langle a^2 \rangle - t^2\langle a^2 : ab \rangle - t^2\langle a^2, ab : b^2 \rangle\right) \\
&\quad = 1 - 3t^3 + t^4 + t^5 - t^2\left(1 - t^2 - t^2(1-t) - t^2(1-t)\right) \\
&\quad = 1 - t^2 - 3t^3 + 4t^4 - t^5
\end{aligned}$$

The above expression is most easily checked by considering a minimal free resolution for the rational quartic: "You take 1 generator of degree 2 and 3 generators of degree 3, and then you subtract off 4 syzygies of degree 4, and add one second syzygy of degree 5."

The Hilbert function of $S/I$ is represented by

$$\frac{1 - t^2 - 3t^3 + 4t^4 - t^5}{(1-t)^4} = \frac{4}{(1-t)^2} + \frac{-3}{(1-t)} - t,$$

so the Hilbert polynomial of $S/I$ is

$$4\binom{d+1}{1} - 3 = 4d + 1,$$

and there is one error term in degree one. The Hilbert polynomial agrees with the Hilbert function in all degrees $d \geq 2$.

## 3. A Quick and Dirty Codimension Algorithm

Often the reason one computes a Hilbert function is to find the dimension, or codimension of a scheme $X$. This information can be determined in less time than the Hilbert function. In this section we present an algorithm for computing the codimension of a monomial ideal.

Dimension is preserved if we replace $I$ by its radical, which is an easy operation: If $I = (x^{B_1}, \ldots, x^{B_m})$, let $x^{C_i}$ be the product of the variables which occur with nonzero exponent in $x^{B_i}$. Then $(x^{C_1}, \ldots, x^{C_m})$ is the radical of $I$.

What is the dimension of $X$ for a radical monomial ideal $I$? In this case, $X$ is a union of coordinate subspaces of $\mathbf{P}^n$, and can be thought of as a simplicial complex embedded in the $n$-simplex of all coordinate subspaces. $I$ is generated by square-free monomials corresponding to the non-cells of $X$.

The codimension of a squarefree monomial ideal $I$ is the minimum size of any subset $X_{in}$ of $x_0, \ldots, x_n$ such that each generator of $I$ is divisible by a variable in $X_{in}$. Such a subset generates an associated prime ideal of $I$ of minimal codimension. There are several ways to organize the search for such subsets. A method which has worked particularly well in our system *Macaulay* is based on the algorithm below.

Briefly, the algorithm is organized in the following way. One loops though each monomial of "monoms", adding a variable of the monomial $A$ to the set $X_{in}$, which contains the variables in the current associated prime, and then calling recursively this routine. On subsequent calls (one for each variable in the monomial $A$), one may assume that the variables already considered do not lie in the associated primes being constructed. These variables are placed in the set $X_{out}$, which contains those variables which cannot be part of the associated prime. The set $X_{unknown}$ contains those variables which have not been placed in $X_{in}$ or $X_{out}$.

At any time, the union of the three sets $X_{in}, X_{out}, X_{unknown}$ is exactly the set $\{x_0, \ldots, x_n\}$. For the purpose of this algorithm, a squarefree monomial is simply a subset of $\{x_0, \ldots, x_n\}$.

**Algorithm 3.1 (codimension of a monomial ideal)**

```
input: A monomial ideal I ⊂ S.
output: a number, the codimension of I.
begin
    compute J := the radical of I.
    rearrange the generators of J in ascending degree lexicographic order
        with x_n > x_{n-1} > ... > x_0, obtaining an ordered
        set of squarefree monomials, named monoms.
    return codim(monoms, {}, {}, {x_0, ..., x_n}).
end.

codim(monoms, X_in, X_out, X_unknown) =
begin
    if monoms = {}
        then return length(X_in).
    let {A} be the first monomial in "monoms".
```

```
    let B = monoms \ {A}.
    if X_in ∩ A ≠
        then return codim(B, X_in, X_out, X_unknown)
    else if A ⊂ X_out
        then return ∞
    else
        write A ∩ X_unknown = {x_{i_1}, ..., x_{i_k}}, for i_1 < ... < i_k.
        return the minimum of the values of
            codim(B, X_in ∪ {x_{i_1}}, X_out, X_unknown\{x_{i_1}}),
            codim(B, X_in ∪ {x_{i_2}}, X_out ∪ {x_{i_1}}, X_unknown\{x_{i_1}, x_{i_2}}),
            ...,
            codim(B, X_in ∪ {x_{i_k}}, X_out ∪ {x_{i_1}, ..., x_{i_{k-1}}},
                X_unknown\{x_{i_1}, ..., x_{i_k}})
end.
```

**Proof.** Let $L$ be the (squarefree) monomial ideal generated by the monomials in "monoms" and the variables in $X_{in}$. We claim that codim(monoms, $X_{in}$, $X_{out}$, $X_{unknown}$) is the minimum of the codimensions of every associated prime ideal of $L$ not containing any variable of $X_{out}$. If the set "monoms" is empty, the codimension of $L$ is the size of $X_{in}$. Otherwise, if $A$ is the first monomial of "monoms", then a minimal associated prime of $L$ must contain a variable occuring in $A$. If some variable of $X_{in}$ occurs in $A$, then $A$ is not a minimal generator, and we can ignore it. If every variable of $A$ is contained in $X_{out}$, then there are no associated primes of $L$ satisfying the above condition. By induction, the call

$$\text{codim}(B,\ X_{in} \cup \{x_{i_j}\},\ X_{out} \cup \{x_{i_1}, \ldots, x_{i_{j-1}}\},\ X_{unknown} \backslash \{x_{i_1}, \ldots, x_{i_j}\})$$

computes the minimum of those associated primes of $L$ which contain $x_{i_j}$, but do not contain any of $x_{i_1}, \ldots, x_{i_{j-1}}$. This establishes the claim, since every associated prime ideal of $L$ not containing any variable of $X_{out}$ is of this form. Therefore the call codim(monoms, {}, {}, $\{x_0, \ldots, x_n\}$) correctly computes the codimension of $J$. □

Recall that the degree lexicographic order first orders monomials by degree, and is the usual lexicographic order on monomials of a given degree. Since each monomial being ordered is squarefree, monomials divisible by the least number of variables occur earliest in the ordered set "monoms".

In an actual implementation, there are a number of improvements that would normally be made. For example, if there were a global variable which gives an upper bound on the codimension, then whenever the size of $X_{in}$ reached this value, one could immediately return this value instead of trying to complete the $X_{in}$ set. Timing data for this algorithm are given in the next section.

## 4. Examples and implementation notes

In this section, we discuss some implementation issues and present some execution times of the Hilbert function algorithms.

Before attempting to implement the algorithms defined in this paper, we had used the algorithm in the *Macaulay* system to compute Hilbert functions. Finding the Hilbert function of a homogeneous ideal involves two steps. A Gröbner basis for the ideal (under any multiplicative order) is first computed, and the Hilbert function of the monomial ideal of initial terms is then found. In small numbers of variables, the Gröbner basis computation takes much longer than the corresponding Hilbert function computation, even if *Macaulay*'s Hilbert function algorithm is used. For more variables, we found the Hilbert function algorithm of *Macaulay* to take much longer than the corresponding Gröbner basis computation. For example, if the number of generators is larger than 1000 or 1500, with 20 or 30 variables, then the execution of *Macaulay*'s Hilbert function algorithm might take several days, or not complete at all. The Hilbert function algorithms defined here, on the other hand, almost always require only a fraction of the time to compute the Gröbner basis; Gröbner bases, rather than Hilbert functions, are now the bottleneck.

For each example below, we present timing data for computing the Gröbner basis of an ideal $I$ using *Macaulay*, for computing the Hilbert function given the initial monomial ideal $J$, and for finding the codimension of the ideal $J$. In two cases, $J$ is only a subset of the initial ideal in($I$). Timing data is presented for three Hilbert function algorithms, named (A), ($A_1$), and (B). The algorithms (A) and (B) correspond to the variants (A) and (B) of Algorithm 2.6. Algorithm variant ($A_1$) is described below. On all but the largest examples, we include the time required by *Macaulay* 3.0's Hilbert function computation. We also include the time required by an implementation of the codimension algorithm of § . The examples below were all run on a Sun microsystems SPARC station 1.

In each of the Hilbert function algorithms, the minimal generators of the monomial ideal $I$ are sorted in ascending lexicographic order. Therefore, if $k$ is the largest index such that $x_k$ divides $x^{A_i}$, then the only variables occuring in $J = (x^{A_1}, \ldots, x^{A_i - 1} : x^{A_i})$ are among $x_0, \ldots, x_{k-1}$. This fact can be used in both the recursion and in the monomial ideal data structure. As we will see in the timing results below, this makes a modest improvement (usually between 5% and 15%) in the performance of the algorithm. Each of the algorithms (A) and (B) incorporates this speedup. In order to measure the resulting performance improvement, we have included timings for variant (A) of Algorithm 2.6 where this feature is not used; each ideal quotient is computed as if all $n + 1$ variables could occur. In the examples and timing data below, this variant is referred to as algorithm ($A_1$).

Overall, the two most important features to implement are: (1) process the monomials in ascending lexicographic order, and (2) use the linear partitioning algorithm, (algorithm (A)). One can delay implementing the general partitioning algorithm (algorithm (B)) until a further performance increase for large problems is needed.

For example, in Example 4.3 below, which is an ideal with 444 generators in 32 variables, *Macaulay* found the Hilbert function in 7 minutes. Modifying algorithm ($A_1$) by first sorting the monomials into reverse lexicographic order rather than lexicographic order requires 19.2 minutes to find the Hilbert series. The time required by algorithm ($A_1$) is 18 *seconds*, a remarkable improvement. We found Algorithm (B) to execute faster on large problems than Algorithm (A), although it uses a large amount of memory. In one case (in Example 4.2,$n = 5$), Algorithm (B) ran out of space at 40 megabytes. For the medium size examples presented here, Algorithms (A) and (B) use about the same

amount of time to find the Hilbert series.

Times are given in minute:second ($mm : ss$) or hour:minute:second ($hh : mm : ss$) format. The Gröbner bases in the following examples were computed using *Macaulay*. The timing data below is accurate to within a few seconds. Space information is accurate to within 64 kilobytes. A "–" entry under space means less than 64 kbytes were used for the given algorithm. The timing information for the codimension algorithm includes the time to input the monomial ideal, as well as the time to compute the square free part of the ideal.

**Example 4.1** Let $M$ be a generic $n$ by $n$ matrix in a polynomial ring $S$ with $n^2$ variables $x_{ij}$, let $I \subset S$ be the ideal generated by the entries of the matrix $M^2$, and let $>$ be the reverse lexicographic order such that $x_{11} > x_{12} > \ldots > x_{1n} > x_{21} > \ldots$. The monomial ideal which is the input to the Hilbert function algorithms is the initial ideal $J = \text{in}(I)$, with respect to this order. The case $n = 3$ takes almost no time, and $n = 4$ is also a small example. For $n = 5$, there are 1372 generators for this ideal: 25 in degree 2, 76 in degree 3, 217 in degree 4, 356 in degree 5, 397 in degree 6, 188 in degree 7 and 113 in degree 8.

The Hilbert series for $S/I$ is given by

$$\begin{aligned}\langle I \rangle = (1-t)^{25} H_{S/I}(t) = \\ = \; & 1 - 25t^2 + 24t^3 + 275t^4 - 576t^5 - 1400t^6 + 6024t^7 + 503t^8 - 51176t^9 \\ & + 179220t^{10} - 393000t^{11} + 671803t^{12} - 963728t^{13} + 1161279t^{14} \\ & - 1105688t^{15} + 687275t^{16} + 18472t^{17} - 740749t^{18} + 1182944t^{19} \\ & - 1222811t^{20} + 960600t^{21} - 600774t^{22} + 303600t^{23} - 123971t^{24} \\ & + 40480t^{25} - 10350t^{26} + 2000t^{27} - 275t^{28} + 24t^{29} - t^{30}\end{aligned}$$

| Example 4.1 ($n = 4$)<br>16 variables, 161 generators<br>26 generators in radical | | |
|---|---|---|
| Algorithm | time | space |
| *Macaulay* Gröbner basis | 0:15 | 126k |
| A | 0:01 | – |
| $A_1$ | 0:01 | – |
| B | 0:01 | – |
| *Macaulay* Hilbert fcn | 0:36 | – |
| codimension = 8 | 0:00 | – |

| Example 4.1 ($n = 5$)<br>25 variables, 1372 generators<br>91 generators in radical | | |
|---|---|---|
| Algorithm | time | space |
| *Macaulay* Gröbner basis | 1:04:15 | 1950k |
| A | 3:06 | 252k |
| $A_1$ | 3:47 | 252k |
| B | 3:08 | 755k |
| *Macaulay* Hilbert fcn | 14:32:24 | 504k |
| codimension = 13 | 0:02 | 126k |

**Example 4.2** Let $M, N$ be generic $n$ by $n$ matrices in a polynomial ring $S$ with $2n^2$ variables $x_{ij}$ and $y_{ij}$. Let $I \subset S$ be the ideal generated by the entries of the product $MN$, and let $>$ be the reverse lexicographic order with $x_{ij} > y_{kl}$, for all possible indices, and such that $x_{11} > x_{12} > \ldots > x_{1n} > x_{21} > \ldots$ and similarly for the $y$ variables.

If $n = 4$, the initial ideal $J = \text{in}(I)$ using this order has 500 elements. For the $n = 5$ example, we use only the Gröbner basis in degrees $\leq 8$. In this case, the resulting monomial ideal $J$ has 4785 generators.

For the $n = 5$ case, the codimension algorithm takes more time than usual due to the fact that $J$ is a radical ideal. This algorithm is still faster than any of the Hilbert function algorithms. The codimension algorithm first sorts the radical of the ideal $J$ in ascending degree lexicographic order. If we change the order in which the monomials are processed to be the *strict* lexicographic order, then the time increases to almost exactly 6 hours. Algorithm (B) finally ran out of space at about 40 megabytes after 2 hours and 52 minutes. We were unable to complete the execution of *Macaulay*'s Hilbert function algorithm on this example.

| Example 4.2 ($n = 4$)<br>32 variables, 500 generators<br>500 generators in radical | | |
|---|---|---|
| Algorithm | time | space |
| *Macaulay* Gröbner basis | 2:59 | 441k |
| A | 0:35 | 126k |
| $A_1$ | 0:42 | 126k |
| B | 0:34 | 504k |
| *Macaulay* Hilbert fcn | 11:01 | 189k |
| codimension = 12 | 0:04 | 126k |

| Example 4.2 ($n = 5$) | | |
|---|---|---|
| Gröbner basis in degrees $\leq 8$ | | |
| 50 variables, 4785 generators | | |
| 4785 generators in radical | | |
| Algorithm | time | space |
| *Macaulay* Gröbner basis | * 5:00:00 | * |
| A | 2:53:23 | 1196k |
| $A_1$ | 3:34:04 | 1196k |
| B | ** | $> 40195k$ |
| *Macaulay* Hilbert fcn | $> 4$ days | ?? |
| codimension = 19 | 26:40 | 1070k |

(*) Time is approximate and space was not measured.

(**) The execution of algorithm (B) ran out of space.

**Example 4.3** Let $S$ be the polynomial ring defined in Example 4.2, and let $>$ be the reverse lexicographic order defined in that example. Let $I$ be the ideal generated by the entries of the matrix $MN - NM$. For $n = 3$, the initial ideal $\text{in}(I)$ has 26 generators, in degrees 2 through 5. The computation of the Hilbert function requires essentially no time (less than .1 seconds, on a Sun Sparc-station 1). For $n = 4$, the entire Gröbner basis has yet to be computed. For this example let $J$ be the first 444 generators of $\text{in}(I)$ computed via *Macaulay*.

| Example 4.3 ($n = 4$ case) | | |
|---|---|---|
| 32 variables, 444 generators | | |
| 248 generators in radical | | |
| Algorithm | time | space |
| *Macaulay* Gröbner basis | * | * |
| A | 0:14 | 126k |
| $A_1$ | 0:18 | 126k |
| B | 0:15 | 441k |
| *Macaulay* Hilbert fcn | 7:17 | 252k |
| codimension = 11 | 0:02 | 126k |

(*) The Gröbner basis computation was not timed.

**Example 4.4** Some of the most intractible ideals known are the Mayr-Meyer ideals. Mayr and Meyer (1982) used these examples to show that the ideal membership problem is double exponential in the number of variables. In this example, we use the simplified equations in Bayer & Stillman (1989, pg. 5). Fix a non-negative integer $n$, and an integer $d \geq 2$. Let

$$S_n = k[z, \{s_i, f_i, b_{i,j}, c_{i,j} : 0 \leq i \leq n, 1 \leq j \leq 4\}]$$

be a polynomial ring in $10(n+1)+1$ variables. Let $I_{n,d} \subset S_n$ be the homogenization with respect to $z$ of the ideal $I_n$ defined in Bayer & Stillman (1989, pg. 5).

When $n = 1$, the polynomial ring above has 21 variables. The monomial order we choose in this case is the product order:

$$s_0 > t_0 > b_{01} > \ldots > b_{04} > c_{01} > \ldots > c_{04} >> s_1 > \ldots > c_{14} >> z,$$

where in each block of variables the (graded) reverse lexicographic order is used. For $d = 2$, $\text{in}(I_{1,2})$ has 444 generators, in degrees ranging from 2 to 11. For $d = 3$, $\text{in}(I_{1,3})$ has 610 generators, in degrees ranging from 2 to 18.

When $n = 2$, the ring has 31 variables. For $d = 2$, we use blocks of 10 variables each (and one final block of 1 variable). In this case, $\text{in}(I)$ has 3204 generators, in degrees ranging from 2 to 32. When $n = 3$, we break up the variables into 6 blocks of 5 variables each, and one final block containing the variable $z$. In this case, $\text{in}(I)$ has 8100 generators in degrees ranging from 2 to 109. There are 324 generators in degree 109 alone.

The timings below indicate that algorithm (B) becomes useful for sufficiently large problems.

| Example 4.4 ($n = 1, d = 2$)<br>21 variables, 444 generators<br>26 generators in radical | | |
|---|---|---|
| Algorithm | time | space |
| *Macaulay* Gröbner basis | 0:30 | 252k |
| A | 0:21 | – |
| $A_1$ | 0:22 | – |
| B | 0:21 | 252k |
| *Macaulay* Hilbert fcn | 12:51 | 189k |
| codimension = 3 | 0:01 | – |

| Example 4.4 ($n = 1, d = 3$)<br>21 variables, 610 generators<br>26 generators in radical | | |
|---|---|---|
| Algorithm | time | space |
| *Macaulay* Gröbner basis | 0:52 | 315k |
| A | 0:48 | 126k |
| $A_1$ | 0:52 | 126k |
| B | 0:45 | 441k |
| *Macaulay* Hilbert fcn | 24:36 | 252k |
| codimension = 3 | 0:02 | – |

| Example 4.4 ($n = 2, d = 2$)<br>31 variables, 3204 generators<br>59 generators in radical | | |
|---|---|---|
| Algorithm | time | space |
| *Macaulay* Gröbner basis | 23:28 | 1321k |
| A | 49:16 | 504k |
| $A_1$ | 52:28 | 504k |
| B | 48:40 | 6039k |
| *Macaulay* Hilbert fcn | ?? | ?? |
| codimension = 5 | 0:02 | 441k |

| Example 4.4 ($n = 2, d = 3$)<br>31 variables, 8100 generators<br>59 generators in radical | | |
|---|---|---|
| Algorithm | time | space |
| *Macaulay* Gröbner basis | 3:52:23 | 3083k |
| A | 7:45:39 | 1636k |
| $A_1$ | 8:58:33 | 1636k |
| B | 6:06:53 | 13902k |
| *Macaulay* Hilbert fcn | ?? | ?? |
| codimension = 5 | 0:04 | 1133k |

## 5. Multi-graded rings

The algorithm we have given for Hilbert functions can be generalized easily to the case when the ring $S = k[x_0, \ldots, x_n]$ is multi-graded.

Fix $m$ weight vectors $w_1, \ldots, w_m \in \mathbf{N}^{n+1}$. The *multi-degree* of a monomial $x^A$ is $(w_1.A, \ldots, w_m.A) \in \mathbf{N}^m$. The $k$-vector space generated by all monomials of multi-degree $(d_1, \ldots, d_m)$ is denoted by $S_{d_1 d_2 \ldots d_m}$. An ideal $I \subset S$ is *multi-graded* if

$$I = \bigoplus_{d_1, \ldots, d_m \in \mathbf{N}} I \cap S_{d_1 d_2 \ldots d_m}.$$

Similarly, one can define multi-graded $S$-modules. The Hilbert series of a multi-graded module $M$ is defined to be the generating function

$$H_M(t_1, \ldots, t_m) := \sum_{d_1, \ldots, d_m \in \mathbf{Z}} \dim M_{d_1 d_2 \ldots d_m} t_1^{d_1} \ldots t_m^{d_m}.$$

The values of the Hilbert function for large values of $d_i$ are not easily encoded into a single polynomial.

As in the homogeneous case, the computation of the Hilbert series of an arbitrary (finitely generated) multi-graded module can be reduced via Gröbner bases to the computation of the Hilbert series of a monomial ideal. If $I$ is a multi-graded ideal, define

$$\langle I \rangle := H_{S/I}(t_1, \ldots, t_m)/H_S(t_1, \ldots, t_m).$$

and for a monomial $x^A$, define

$$t^{|A|} := t_1^{w_1 \cdot A} \ldots t_m^{w_m \cdot A}.$$

With these new definitions, Proposition 2.2 holds as follows.

**Proposition 5.1** *Let $I$ be a monomial ideal of the multi-graded ring $S$. Write $I = (J, x^A)$ for a monomial ideal $J$. Then*

*(a)* $\langle x^A \rangle = 1 - t^{|A|}$;

*(b)* $\langle J \cap (x^A) \rangle = 1 - t^{|A|} + t^{|A|} \langle J : x^A \rangle$;

*(c)* $\langle I \rangle = \langle J \rangle - t^{|A|} \langle J : x^A \rangle$.

Similarly, Proposition 2.4 generalizes to the multi-graded case as well. Using the new definitions for $\langle J \rangle$ and $t^{|A|}$, Algorithm 2.6 can be used exactly as stated, except that in variant (A), $\langle J \rangle$ should be set to

$$\langle J \rangle := (1 - t^{|x_{j_1}|}) \dots (1 - t^{|x_{j_k}|}) \langle J_1 \rangle$$

All that remains is to identify the Hilbert series of $S$. This is easy: the Hilbert series of $S$ is

$$H_S(t_1, \dots, t_m) = \frac{1}{(1 - t^{|x_0|})(1 - t^{|x_1|}) \dots (1 - t^{|x_n|})}.$$

**Example 5.2** The ideal $I \subset S$ of § is bigraded, with respect to the gradings

$$w_1 = (1, 1, 1, 1)$$

and

$$w_2 = (1, 4, 3, 0).$$

The numerator $\langle I \rangle$ of the multi-graded Hilbert series of $I$, computed via variant (A), is

$$\begin{aligned}
&\langle a^3, a^2b, ab^2, ac, b^3d \rangle \\
&= \langle a^3 \rangle - t_1^3 t_2^6 \langle a^3 : a^2b \rangle - t_1^3 t_2^9 \langle a^3, a^2b : ab^2 \rangle - t_1^2 t_2^4 \langle a^3, a^2b, ab^2 : ac \rangle \\
&\quad - t_1^4 t_2^{12} \langle a^3, a^2b, ab^2, ac : b^3d \rangle \\
&= 1 - t_1^3 t_2^3 - t_1^3 t_2^6 \langle a \rangle - t_1^3 t_2^9 \langle a \rangle - t_1^2 t_2^4 \langle a^2, ab, b^2 \rangle - t_1^4 t_2^{12} \langle a \rangle \\
&= 1 - t_1^3 t_2^3 - t_1^3 t_2^6 \langle a \rangle - t_1^3 t_2^9 \langle a \rangle \\
&\quad - t_1^2 t_2^4 \left( \langle a^2 \rangle - t_1^2 t_2^5 \langle a^2 : ab \rangle - t_1^2 t_2^8 \langle a^2, ab : b^2 \rangle \right) - t_1^4 t_2^{12} \langle a \rangle \\
&= 1 - t_1^3 t_2^3 - t_1^3 t_2^6 (1 - t_1 t_2) - t_1^3 t_2^9 (1 - t_1 t_2) \\
&\quad - t_1^2 t_2^4 \left( 1 - t_1^2 t_2^2 - t_1^2 t_2^5 (1 - t_1 t_2) - t_1^2 t_2^8 (1 - t_1 t_2) \right) - t_1^4 t_2^{12} (1 - t_1 t_2) \\
&= 1 - t_1^2 t_2^4 - t_1^3 t_2^3 - t_1^3 t_2^6 - t_1^3 t_2^9 + t_1^4 t_2^6 + t_1^4 t_2^7 + t_1^4 t_2^9 + t_1^4 t_2^{10} - t_1^5 t_2^{10}
\end{aligned}$$

As in the homogeneous case, this can be checked using finite free resolutions: "Take generators of degrees (2,4), (3,3), (3,6), and (3,9), and then subtract off 4 syzygies of degrees (4,6), (4,7), (4,9), and (4,10), and add one second syzygy of degree (5,10)."

## 6. Applications of Hilbert functions

The most basic application of Hilbert functions is to find numerical invariants of a subscheme $X \subset \mathbf{P}^n$. For example, if $I$ is the homogeneous ideal of $X$, and if the Laurent expansion in $(1 - t)$ of the Hilbert series of $S/I$ is

$$\frac{a_{-r-1}}{(1-t)^{r+1}} + \dots + \frac{a_{-1}}{(1-t)} + a_0 + \dots + a_s t^s,$$

then $\dim X = r$, $\deg X = a_{-r-1}$, and the arithmetic genus $p_a(X)$ of $X$ is $p_a(X) = (-1)^r(a_{-r-1} + a_{-r} + \ldots + a_{-1} - 1)$.

Hilbert functions are also useful as an algorithmic tool. For example, Hilbert functions can be used to determine if two schemes are equal, or "nearly" equal, in the following sense. Given two (homogeneous) ideals $I \subset J$ of $S$, $I$ and $J$ are called *scheme-theoretically* equal if they define the same projective scheme in $\mathbf{P}^n$. Equivalently, $I$ and $J$ are scheme-theoretically equal if they are equal in large degrees. In this case, $I$ and $J$ have exactly the same primary components, except that $I$ has (possibly) an additional embedded component supported at the irrelevant maximal ideal.

More generally, we say that $I$ and $J$ are *equal in codimension* $< c$ if they have the same primary components in codimension $< c$: i.e. the localizations $I_P$ and $J_P$ are equal, for all prime ideals $P$ of codimension $< c$. Similarly, we say that $I$ and $J$ are equal in dimension $> d$ if $I_P = J_P$ for every prime ideal of $S$ with $\dim S/P > d$. Two ideals $I$ and $J$ of $S$ are scheme-theoretically equal if they are equal in codimension $< n + 1$, or equivalently if they are equal in dimension $> 0$.

One can use Hilbert polynomials to detect equality of two ideals in codimension $< c$, as long as $c \leq n + 1$. The following (equivalent) test uses numerators of Hilbert series instead, so that equality of ideals can also be detected.

**Proposition 6.1** *If $I \subset J \subset S$ are homogeneous ideals, then $I$ and $J$ are equal in codimension $< c$ if and only if $\langle I \rangle - \langle J \rangle$ is divisible by $(1-t)^c$.*

*In terms of Hilbert polynomials, if $h_{S/I}$ and $h_{S/J}$ are the Hilbert polynomials of $S/I$ and $S/J$ respectively, and if $d \geq 1$, then $I = J$ in (Krull) dimension $> d$ if and only if $h_{S/I}(z) - h_{S/J}(z)$ is a polynomial of degree $< d$ in $z$.*

**Proof.** $I \subset J$ are equal in codimension $< c$ if and only if $\dim J/I < n + 1 - c$, which is true if and only if the Hilbert series, $H_{J/I}(t)$ of $J/I$ has a pole of order $< n + 1 - c$. But

$$H_{J/I}(t) = H_{S/I} - H_{S/J} = \frac{\langle I \rangle - \langle J \rangle}{(1-t)^{n+1}}.$$

Therefore $H_{J/I}$ has a pole of order $< n + 1 - c$ if and only if $\langle I \rangle - \langle J \rangle$ is divisible by $(1-t)^c$. □

It is also possible to use Hilbert functions to test for equality of inhomogeneous ideals. Let $>$ be a degree preserving multiplicative order on the monomials of $R = k[x_1, \ldots, x_n]$. Let $I \subset R$ be a (not necessarily homogeneous) ideal. Define $\langle I \rangle := \langle \mathrm{in}(I) \rangle$. This polynomial does not depend on the particular degree preserving monomial order $>$.

**Proposition 6.2** *Let $I \subset J \subset k[x_1, \ldots, x_n]$ be ideals, and define $\langle I \rangle$ and $\langle J \rangle$ as above. The ideals $I$ and $J$ are equal in codimension $< c$ if and only if $\langle I \rangle - \langle J \rangle$ is divisible by $(1-t)^c$.*

**Proof.** Let $I^h$ be the ideal generated by $\{f^h \in S : f \in I\}$, where $f^h$ is the homogenization of the polynomial $f$ with respect to the variable $x_0$. $I^h$ is the homogenization of $I$.

Extend the order $>$ of the monomials of $R$ to the monomials of $S$ by setting $x_0^a x^A > x_0^b x^B$ if and only if $x^A > x^B$ or $x^A = x^B$ and $a > b$. It is well-known (and easy) that if $\{g_1, \ldots, g_s\}$ is a Gröbner basis of $I$ with respect to $>$, then $\{g_1^h, \ldots, g_s^h\}$ is a Gröbner basis for $I^h$. Furthermore, since the lead monomial of $g_i$ always has the highest degree of any monomial in $g_i$, $\text{in}(g_i) = \text{in}(g_i^h)$. Therefore $\langle I^h \rangle = \langle \text{in}(I^h) \rangle = \langle \text{in}(I) \rangle$. Notice that, in particular, $\langle I \rangle$ is well-defined. By the definition of $\langle I \rangle$, we see that $\langle I^h \rangle = \langle I \rangle$, and similarly for $J$.

If

$$I = Q_1 \cap \ldots \cap Q_p$$

is an irredundant primary decomposition of $I$, then

$$I^h = Q_1^h \cap \ldots \cap Q_p^h$$

is an irredundant primary decomposition of $I^h$; the components of $I$ and $I^h$ correspond in a 1-1 manner. Also, the codimension of $I$ in $R$ is the same as the codimension of $I^h$ in $S$. Therefore, $I$ and $J$ have the same primary components in codimension $< c$ if and only if $I^h$ and $J^h$ have the same primary components in codimension $< c$. By the above proposition, this holds if and only if $\langle I^h \rangle - \langle J^h \rangle$ is divisible by $(1-t)^c$, which in turn holds if and only if $\langle I \rangle - \langle J \rangle$ is divisible by $(1-t)^c$. □

Notice also that the codimension of $I$ in $R$ is exactly the codimension of the monomial ideal $\text{in}(I)$ of $R$.

The first named author was partially supported by the Alfred P. Sloan Foundation, by ONR contract N00014-87-K0214, and by NSF grant DMS-90-06116. The second author was partially supported by the U.S. Army Research Office through the Mathematical Sciences Institute of Cornell University, and by NSF grants CCR-89-01061 and DMS-88-02276. We are grateful for their support.

## References

Atiyah, M. F.,MacDonald, I. G. (1969). *Introduction to Commutative Algebra*, Addison-Wesley Series in Mathematics.

Bayer, D. (1982). *The division algorithm and the Hilbert scheme*, Ph.D. thesis, Harvard University. Order number 82-22588, University Microfilms International, 300 N. Zeeb Rd., Ann Arbor, MI 48106.

Bayer, D., Galligo, A., Stillman, M. (1992). *Computation of Primary Decompositions*, in preparation.

Bayer, D., Stillman, M. (1987a). *A criterion for detecting* m*-regularity*, Invent. Math. **87**, 1-11.

Bayer, D., Stillman, M. (1987b). *A theorem on refining division orders by the reverse lexicographic order*, Duke Math. J. Vol. 55, No. 2.

Bayer, D., Stillman, M. (1989). *On the complexity of computing syzygies*, in "Computational Aspects of Commutative Algebra", ed. L. Robbiano, Academic Press, San Diego.

Bayer, D., Stillman, M. (1982-1992). *Macaulay: A system for computation in algebraic geometry and commutative algebra*, available for Unix and Macintosh computers. Contact the authors, or `ftp 128.103.28.10, Name: ftp, Password: any, cd Macaulay, binary, get M3.tar, quit, tar xf M3.tar`.

Bayer, D., Stillman, M. (1992). *Upper bounds on the betti numbers of graded ideals*, in preparation.

Buchberger, B. (1965). *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*, Ph.D. Thesis, Universität Innsbruck.

Buchberger, B. (1985). *Gröbner bases: An algorithmic method in polynomial ideal theory*, in: (N.K. Bose, ed) "Multidimensional Systems Theory," D. Reidel Publishing Co., pp. 184-232.

Eliahou, S., Kervaire, M. (1990). *Minimal resolutions of some monomial ideals*, J. Algebra **129**, 1-25.

Galligo, A. (1974). *A propos du théorème de préparation de Weierstrass, Fonctions de Plusiers Variables Complexes*, Lecture Notes in Math. **409**, 543-579.

Garey, M., Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, San Francisco. ISBN 0-7167-1044-7, 0-7167-1045-5 pbk.

Karp, R.M. (1972). *Reducibility among combinatorial problems*, in R.E. Miller and J.W.Thatcher (eds), "Complexity of computer computations", Plenum Press, New York, 85-103.

Macaulay, F.S. (1927). *Some properties of enumeration in the theory of modular systems*, Proc. London Math. Soc. **26**, 531-555.

Mayr, E., Meyer, A. (1982). *The complexity of the word problem for commutative semigroups and polynomial ideals*, Adv. Math **46**, 305-329.

Mora, F., Möller, H.M. (1983). *The computation of the Hilbert function*, Lecture Notes in Computer Science **162** (EUROCAL '83), 157-167.

Stanley, R. (1978). *Hilbert functions of graded algebras*, Adv. in Math. **28**, 57-83.